Übungen zum Kapitel 6

Daten laden, speichern und Dateiformate

Erstellt und überarbeitet: armin.baenziger@zhaw.ch, 20. Januar 2020

In [1]:
```
%autosave 0
```

```
Autosave disabled
```

(A.1) Laden Sie NumPy und Pandas mit den üblichen Abkürzungen.

In [2]:
```
import numpy as np
import pandas as pd
```

(A.2) Laden Sie die Daten der Datei `drinksbycountry.csv` in das DataFrame `drinks` . Die Datei befindet sich im Ordner "weitere_Daten". Lesen Sie danach die ersten 5 Zeilen des DataFrames `drinks` aus.

In [3]:
```
drinks = pd.read_csv('../weitere_Daten/drinksbycountry.csv')
drinks.head()
```

Out[3]:

	country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
0	Afghanistan	0	0	0	0.0	Asia
1	Albania	89	132	54	4.9	Europe
2	Algeria	25	0	14	0.7	Africa
3	Andorra	245	138	312	12.4	Europe
4	Angola	217	57	45	5.9	Africa

(A.3) Wiederholen Sie **(A.2)**, wobei Sie nun *beim Einlesen* die Spalte/Varibale `country` als Index festlegen. Dies geschieht mit dem zusätzlichen Funktionsargument `index_col='country'` .

In [4]:
```
drinks = pd.read_csv('../weitere_Daten/drinksbycountry.csv',
                     index_col='country')
drinks.head()
```

Out[4]:

country	beer_servings	spirit_servings	wine_servings	total_litres_of_pure_alcohol	continent
Afghanistan	0	0	0	0.0	Asia
Albania	89	132	54	4.9	Europe
Algeria	25	0	14	0.7	Africa
Andorra	245	138	312	12.4	Europe
Angola	217	57	45	5.9	Africa

(A.4) Löschen Sie die Spalten/Variablen `beer_servings`, `spirit_servings` und `wine_servings` aus dem DataFrame `drinks`.

In [5]:
```
## Es gibt verschiedene Lösungsmöglichkeiten:
# drinks.drop(['beer_servings', 'spirit_servings', 'wine_servings'],
# axis=1, inplace=True)
## Oder mit:
# del drinks['beer_servings']  # usw.
## Oder aber durch umkopieren:
drinks = drinks.loc[:, ['total_litres_of_pure_alcohol', 'continent']]
drinks.head()
## Schliesslich hätte man die Datei auch nochmals mit dem Argment
# "usecols" einlesen können.
```

Out[5]:

country	total_litres_of_pure_alcohol	continent
Afghanistan	0.0	Asia
Albania	4.9	Europe
Algeria	0.7	Africa
Andorra	12.4	Europe
Angola	5.9	Africa

Zur Vereinfachung wählen wir kürzere Variablennamen:

In [6]:
```
drinks.rename(columns={'total_litres_of_pure_alcohol': 'alcohol'},
              inplace=True)
drinks.head()
```

Out[6]:

country	alcohol	continent
Afghanistan	0.0	Asia
Albania	4.9	Europe
Algeria	0.7	Africa
Andorra	12.4	Europe
Angola	5.9	Africa

(A.5) Wie hoch ist der höchste Konsum an Alkohol (Liter pro Jahr)?

In [7]:
```
drinks.alcohol.max()
```

Out[7]: `14.4`

(A.6) Sortieren Sie das DataFrame `drinks` nach Alkoholkunsum in *absteigender* Reihenfolge. Tipp: Methode `sort_values`.
Lesen Sie dann die 5 Länder mit dem grössten Alkoholkonsum aus.

In [8]:
```
drinks.sort_values('alcohol', ascending=False).head()
```

Out[8]:

country	alcohol	continent
Belarus	14.4	Europe
Lithuania	12.9	Europe
Andorra	12.4	Europe
Grenada	11.9	North America
Czech Republic	11.8	Europe

A.7) Wie viele Länder gibt es pro Kontinent (im Datensatz)?

In [9]:
```
drinks.continent.value_counts()
```

Out[9]:
```
Africa           53
Europe           45
Asia             44
North America    23
Oceania          16
South America    12
Name: continent, dtype: int64
```

(A.8) Speichern Sie die Daten des DataFrames `drinks` in einer CSV-Datei mit Name `temp.csv` auf Ihren Datenträger.

In [10]:
```
drinks.to_csv('temp.csv')
```

(A.9) Speichern Sie die ersten 5 Zeilen von `drinks` in einer `csv` -Datei mit Semikolon (statt Komma) als Trennzeichen. (Dateiname wiederum `temp.csv`)

In [11]:
```
drinks.head().to_csv('temp.csv', sep=';')
```

(A.10) Prüfen Sie mit dem Magic-Command `%load` , ob der letzte Task funktioniert hat, also mit `%load temp.csv` . Alternativ können Sie folgende Anweisung versuchen:

- auf Linux-/Mac-Systemen: `!cat temp.csv`
- auf Windows-Systemen: `!type temp.csv`

In [12]:
```
!type temp.csv
```

```
country;alcohol;continent
Afghanistan;0.0;Asia
Albania;4.9;Europe
Algeria;0.7;Africa
Andorra;12.4;Europe
Angola;5.9;Africa
```

(A.11) Importieren Sie die Bibliothek `os` und löschen Sie dann die Datei `temp.csv` wieder vom Datenträger mit der Anweisung `os.remove('temp.csv')` . (Alternativ können Sie die Datei manuell im Verzeichnis löschen.)

In [13]:
```
import os
os.remove('temp.csv')
```

(B.1) Betrachten Sie den Inhalt der Datei `Auto.csv` im Ordner `weitere_Daten` in einem Texteditor. Laden Sie dann die Daten mit dem korrekten `sep` -Argument ins DataFrame `Auto` .

In [14]:
```
Auto = pd.read_csv('../weitere_Daten/Auto.csv', sep=';')
Auto.head()
```

Out[14]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

(B.2) Berechnen Sie die Korrelation (nach Bravais-Pearson) zwischen dem Gewicht von Autos (`weight`) und der Treibstoffeffizienz (`mpg`).

In [15]:
```
Auto.weight.corr(Auto.mpg)
```

Out[15]:
```
-0.8322442148315754
```

(B.3) Erstellen Sie eine absolute Häufigkeitsverteilung der Variable `origin` . (1 steht für USA, 2 für Europa und 3 für Japan.)

In [16]:
```
Auto.origin.value_counts()
```

Out[16]:
```
1    245
3     79
2     68
Name: origin, dtype: int64
```

(B.4) Erstellen Sie eine *relative* Häufigkeitsverteilung der Anzahl Zylinder im Datensatz. Tipp: Argument `normalize=True` verwenden.
Versuchen Sie, die Tabelle nach der Anzahl Zylinder (und nicht der Häufigkeit) zu sortieren.

In [17]:
```
Auto.cylinders.value_counts(normalize=True, sort=False)
```

Out[17]:
```
3    0.010204
4    0.507653
5    0.007653
6    0.211735
8    0.262755
Name: cylinders, dtype: float64
```

Bevor wir weiterfahren erstellen wir ein DataFrame mit simulierten Daten:

In [18]:
```
np.random.seed(125)
n = 100

dflohn = pd.DataFrame({'Person': (range(1,n+1)),
  'Lohn'      : np.random.lognormal(mean=8.6, sigma=0.4, size=n).astype(int),
  'Geschlecht': np.random.choice(['w','m'], size=n, replace=True),
  'Alter'     : np.random.randint(18, 66, n),
  'Zivilstand': np.random.choice(['l','v','g','vw'], size=n, replace=True)})

dflohn.head(5)
```

Out[18]:

	Person	Lohn	Geschlecht	Alter	Zivilstand
0	1	4107	m	40	g
1	2	5454	m	47	vw
2	3	3719	m	41	g
3	4	6194	m	18	v
4	5	6161	m	27	v

(C.1) Setzten Sie die Spalte `Person` als Index fest. Verwenden Sie hierzu die Anweisung `dflohn.set_index('Person', inplace=True)` .

In [19]:
```
dflohn.set_index('Person', inplace=True)
dflohn.head()
```

Out[19]:

	Lohn	Geschlecht	Alter	Zivilstand
Person				
1	4107	m	40	g
2	5454	m	47	vw
3	3719	m	41	g
4	6194	m	18	v
5	6161	m	27	v

(C.2) Ersetzen Sie den *Lohn* von *Person 5* mit dem Fehlwert `NaN` (`None` , oder `np.nan`).

In [20]:
```
dflohn.loc[5, 'Lohn'] = None
dflohn.head()
```

Out[20]:

	Lohn	Geschlecht	Alter	Zivilstand
Person				
1	4107.0	m	40	g
2	5454.0	m	47	vw
3	3719.0	m	41	g
4	6194.0	m	18	v
5	NaN	m	27	v

(C.3) Speichern Sie die Daten des DataFrames `dflohn` in einer Datei im "Pickle-Format" mit Name `dflohn.pkl` *im Verzeichnis* `weitere_Daten` auf Ihrem Datenträger. **Wir werden diesen simulierten Datensatz später im Kurs nutzen!**

In [21]:
```
dflohn.to_pickle('../weitere_Daten/dflohn.pkl')
```

(D.1) Mit Pandas können auch Daten von (anderen) wichtigen Statistikpakten, wie R oder Stata, gelesen werden. Im Verzeichnis `weitere_Daten` befindet sich die Stata-Datei `Chang.dta`, welche Daten aus einer Veröffentlichung aus dem Jahre 2018 enthält. Laden Sie die Daten in ein DataFrame mit Name `chang`. Die Funktion dafür lautet `pd.read_stata()`.

Chang, Y., Hong, J.H. und Karabarbounis, M. (2018). *Labor Market Uncertainty and Portfolio Choice Puzzles.* American Economic Journal: Macroeconomics, 10(2), S. 222–262.

In [22]:
```
chang = pd.read_stata('../weitere_Daten/Chang.dta')
chang.head()
```

Out[22]:

	year	wgt	wage_risk	age	educ	income	children	marital	credit_cards_debt	other_consur
0	1998.0	165.988480	0.0529	65.0	2.0	710000.0	4.0	1.0	0.0	
1	1998.0	1663.196899	0.0529	65.0	2.0	467000.0	4.0	1.0	0.0	
2	1998.0	1748.509155	0.0529	65.0	2.0	751000.0	4.0	1.0	0.0	
3	1998.0	1713.679321	0.0529	65.0	2.0	333000.0	4.0	1.0	0.0	
4	1998.0	1847.267334	0.0529	65.0	2.0	393000.0	4.0	1.0	0.0	

5 rows × 27 columns

Die nächste Zelle ändert den Datentyp der Variable `children` zu `int` (Ganzzahl).

In [23]:
```
chang['children'] = chang.children.astype(int)
chang.head()
```

Out[23]:

	year	wgt	wage_risk	age	educ	income	children	marital	credit_cards_debt	other_consur
0	1998.0	165.988480	0.0529	65.0	2.0	710000.0	4	1.0	0.0	
1	1998.0	1663.196899	0.0529	65.0	2.0	467000.0	4	1.0	0.0	
2	1998.0	1748.509155	0.0529	65.0	2.0	751000.0	4	1.0	0.0	
3	1998.0	1713.679321	0.0529	65.0	2.0	333000.0	4	1.0	0.0	
4	1998.0	1847.267334	0.0529	65.0	2.0	393000.0	4	1.0	0.0	

5 rows × 27 columns

(D.2) Erstellen Sie eine (absolute) Häufigkeitstabelle der Variable `children`.

In [24]:
```
chang.children.value_counts()
```

Out[24]:
```
0     37431
2     16748
1     11953
3     10783
4      5713
5      2824
6      1402
7       691
8       345
10      300
9       225
Name: children, dtype: int64
```

(D.3) Erstellen Sie eine *relative* Häufigkeitstabelle der Variable `children`.

In [25]:
```
chang.children.value_counts(normalize=True)
```

Out[25]:
```
0     0.423356
2     0.189425
1     0.135192
3     0.121959
4     0.064616
5     0.031940
6     0.015857
7     0.007815
8     0.003902
10    0.003393
9     0.002545
Name: children, dtype: float64
```

(D.3) Erstellen Sie nochmals eine *relative* Häufigkeitstabelle der Variable `children`. Sortieren Sie aber nach der Anzahl Kinder (und nicht der Häufigkeit). Speichern Sie das Ergebnis im Objekt `tabelle`.

In [26]:
```
tabelle = chang.children.value_counts(normalize=True).sort_index()
tabelle
```

Out[26]:
```
0     0.423356
1     0.135192
2     0.189425
3     0.121959
4     0.064616
5     0.031940
6     0.015857
7     0.007815
8     0.003902
9     0.002545
10    0.003393
Name: children, dtype: float64
```

Hinweis: Wenn es die Zeit erlaubt, werden wir die Daten von Chang et al. später genauer untersuchen.

Ende der Übung